Introduction to Databases, ITU, Fall 2005

Rasmus Pagh

November 2, 2005

Exercises on November 7

1. **Exam problem from Databasesystemer, June 2005 (15 %)**
   Consider the following SQL queries:

```
1:    SELECT speciale
      FROM Behandler
      WHERE postnr=1000 AND sygdom='hundegalskab';

2:    SELECT speciale
      FROM Behandler
      WHERE postnr=1000 AND id IN (SELECT id
                                   FROM Behandler
                                   WHERE sygdom='hundegalskab');
```

   The two queries refer to the attributes `speciale`, `postnr`, `sygdom` and `id`. All attributes are selective in the sense that only a small fraction of the tuples will have a particular value on any given attribute. The attribute `id` is even a key for the relation (but not declared as primary key).

   To improve the running time of the queries, it is considered to create one or more indexes. This problem is concerned with evaluating different possibilities for indexing.

   (a) Evaluate for each of the mentioned attributes the effect of having an index on this attribute as the *only* index. State which of the two queries (if any) the index would make run faster. Which possibility (or possibilities) would be best, with regards to making both queries fast? Argue for your answer.

   (b) Consider each of the following four possibilities for forming a composite index:

   - `CREATE INDEX idx1 ON Behandler(postnr,sygdom);`
   - `CREATE INDEX idx2 ON Behandler(sygdom,postnr);`
   - `CREATE INDEX idx3 ON Behandler(id,postnr);`
   - `CREATE INDEX idx4 ON Behandler(postnr,id);`

   Again, evaluate for each of the mentioned indexes the effect of having it as the *only* index. State which of the two queries (if any) the index would make run faster.

   (c) Propose a combination of two indexes that would make the queries run as fast as possible. Argue for your choice.

2. The following exercises refer to the examples belonging to the lecture on "database efficiency", which can be found on the course home page. In Oracle, perform the preliminaries to enable autotrace, and get the sample data, as in the examples.

3. Run these example statements (you can copy and paste from the PDF file):

```
SELECT count(*) FROM projects WHERE last2='Gearloose' and last3='Rabbit';
INSERT INTO projects values ('Rasmus','Pagh',11,'Toger','Norgaard',10,'Lars','Bengtsson',10);
CREATE INDEX groupmembers23 on projects (last2,last3,first2,first3);
INSERT INTO projects values ('Rasmus','Pagh',13,'Toger','Norgaard',11,'Lars','Bengtsson',11);
SELECT count(*) FROM projects WHERE last2='Gearloose' and last3='Rabbit';
SELECT count(*) FROM projects WHERE last2='Gearloose';
SELECT count(*) FROM projects WHERE last3='Rabbit';
```

   For each query notice the kind of scan used, as shown in the execution plan. Try to compare and explain the changes in the number of "db block gets" for similar operations. This is the number of times the DBMS accesses a block of data, either part of an index or part of a relation. Not all of these give rise to physical gets, as some data is stored, "cached", in main memory. In principle the time spent should closely follow the number of physical reads reported – however, in the current Oracle installation at ITU there seems to be an additional cache unknown to the DBMS, which means that some reads thought to be physical by Oracle are really main memory reads.

4. Run these example statements:

```
SELECT count(*) FROM projects WHERE last1='Gearloose' and last2='Rabbit';
SELECT max(grade2) FROM projects WHERE last1='Gearloose' and last2='Rabbit';
```

5. Compare the speed of updates to the database with and without an index, as follows:

   - Write `commit;` to commit all previous changes.
   - Delete all projects with `grade1` $<$ `6`. Note the time in seconds and the number of db gets / physical reads.
   - Write `rollback;` to get back to the state before the deletion.
   - Create an index on `grade2`.
   - Again delete all projects with `grade1` $<$ `6`. Again note the time in seconds and the number of db gets / physical reads.